

# МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

# «ЮЖНО-РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (НПИ) им. М.И. Платова»**

# ФАКУЛЬТЕТ Информационных технологий и управления КАФЕДРА Информационные и измерительные системы и технологии НАПРАВЛЕНИЕ 09.03.03 – «Прикладная Информатика»

КУРСОВАЯ РАБОТА

# На тему: Web-сайт подразделения оценки профессиональных компетенций сотрудников предприятия

**По дисциплине: разработка приложений для Интернет По дисциплине: Разработка приложений**

## Выполнил студентка 3 курса, группы 3-2Б Сур Дарья Владимировна

Фамилия, имя, отчество

## Принял доцент Панфилов Александр Николаевич

Должность, звание Фамилия, имя, отчество

## Работа принята « \_» г.

Подпись

## Новочеркасск 2020 г.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_bookmark0)

[ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ИНТЕРНЕТПРИЛОЖЕНИЯ 4](#_bookmark1)

* 1. [Описание информационных и бизнес процессов в рамках предметной](#_bookmark2) [области Интернет – приложения 4](#_bookmark2)
  2. [Построение модели Интернет- приложения 4](#_bookmark3)

1. [ПРОЕКТИРОВАНИЕ WEB-САЙТА 6](#_bookmark4)
   1. [Функциональная структура и модули сайта 6](#_bookmark5)
   2. [Дизайн сайта 7](#_bookmark6)
2. [ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ САЙТА 9](#_bookmark7)
   1. [Концептуальное проектирование базы данных 9](#_bookmark8)
   2. [Логическое проектирование базы данных 10](#_bookmark9)
   3. [Физическая модель базы данных 12](#_bookmark10)
3. [ПРОГРАММНАЯ РЕАЛИЗАЦИЯ САЙТА 15](#_bookmark11)
   1. [Функциональная структура и модули сайта 15](#_bookmark12)
   2. [Описание пользовательского интерфейса 16](#_bookmark13)
   3. [Реализации админ-панели 17](#_bookmark14)
4. [ОЦЕНКА ЭФФЕКТИВНОСТИ И ТЕСТИРОВАНИЕ САЙТА 19](#_bookmark15)
   1. [Тестирование сайта 19](#_bookmark16)

[ЗАКЛЮЧЕНИЕ 23](#_bookmark17)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 24](#_bookmark18)

[Приложение А. Техническое задание на разработку Web-Сайта 25](#_bookmark19)

[Приложение Б. Каскадные таблицы стилей 29](#_bookmark20)

[Приложение В. Тексты программ 31](#_bookmark21)

# ВВЕДЕНИЕ

## Целью данной курсовой работы является создание Web-сайта подразделения оценки профессиональных компетенций сотрудников предприятия. Для достижения поставленной цели были определены следующие задачи:

* описать информационныепроцессы в рамках предметной области;

## построить модель Интернет – приложения;

* спроектировать функциональную структуру и модули сайта;

## спроектировать дизайн сайта;

* описать информационное обеспечение сайта;

## выполнить программную реализацию сайта.

## ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ИНТЕРНЕТПРИЛОЖЕНИЯ

## Задачами разработки Web-сайта подразделения оценки профессиональных компетенций сотрудников предприятия являются предоставление удобного и быстрого способа получения исчерпывающей информации о квалификации сотрудников, оценки их способностей и уровне знаний.

* 1. Описание информационных и бизнес процессов в рамках предметной

## области Интернет – приложения

Созданный мною Web-сайт будет являться отличным средством для

## классификации рабочих кадров и структурирования информации о них. Обусловлено данное удобство тем, что с помощью этого приложения данные легко заносить в базу данных, для их упорядоченного хранения, а так же сайт поможет дать актуальную информацию о каждом из работников, а именно, об уровне его знаний для той или иной должности. Такая проверка

сотрудников каждого из предприятий поможет повысить работоспособность предприятия.

## Построение модели Интернет- приложения Разрабатываемый web-сайт должен содержать следующие страницы:

* главная страница;

## тесты;

* должности;

## пользователи;

* о проекте;

## вход;

-регистрация.

## Web – страница “Главная” содержит всю главную и необходимую информацию о сайте, для того, чтобы пользователь понимал как далее работать с этой программной платформой.

Web – страница “Тесты” содержать список всех доступных к прохождению тестов.

## Web – страница “Должности” будет содержать список всех должностей, которые соответствуют каждому из сотрудников.

Web – страница “Пользователи” содержит список всех сотрудников

## предприятия, которые занесены в базу данных, идентифицированы и имеют доступ к тестам.

Web – страница “О проекте” содержит информацию о сайте.

## Web – страница “Регистрация” необходима для регистрации пользователей на сайте.

Web – страница “Вход” необходима для авторизации уже зарегестрированных пользователей на сайте.

## ПРОЕКТИРОВАНИЕ WEB-САЙТА

В процессе проектирования *Web*-сайта, необходимо выполнить следующие задачи:

## построить функциональную структуру;

* + описать модули *Web*-сайта;

## описать дизайн *Web*-сайта.

Web-сайт подразделения оценки профессиональных компетенций сотрудников предприятия должен иметь следующий функционал:

## вывод из базы информации о сотрудниках

* регистрация и авторизация пользователя;

## онлайн тестирование;

* просмотр пройденных тестов и оценок пользователя.

## Требования к программируемой среде и средствам разработки следующие:

* формат представления Web-документов – HTML версии 4.0 и выше;

## язык разработки серверных приложений – JavaScript;

* операционная система – Ubuntu 20.04;

## сервер баз данных – PostgreSQL.

* 1. Функциональная структура и модули сайта К основным функциям системы относится:

## регистрация и авторизация пользователя;

1. вывод списка тестов;

## вывод списка категорий тестов;

1. вывод списка сотрудников

## вывод списка категорий тестов;

1. вывод контактной информации;

## вывод информации о тесте;

1. вывод информации о сотруднике

## подсчет результатов тестирования.

Функциональная структура описывает основные функции сайта, которые необходимо реализовать в ходе выполнения курсовой работы. Для проектирования функциональной модели сайта использовался программный продукт Microsoft Visio 2016. Функциональная модель сайта представлена на рисунке 2.1.



## Рисунок 2.1 – Функциональная модель сайта

## Дизайн сайта

## Структура Web-сайта представлена на рисунке 2.2. *Web*-сайта должен содержать в своей структуре размещения блоков следующие элементы:

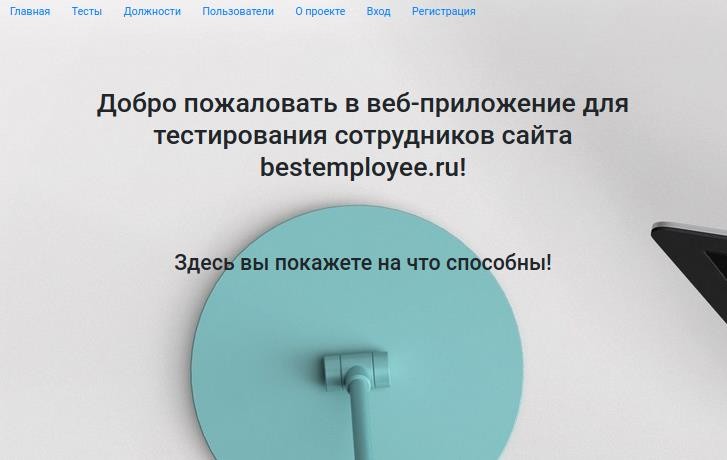
1. навигационное меню, которое содержит ссылки на все основные разделы сайта;

## вспомогательная панель навигации, содержащая якорные ссылки для перехода к конкретной категории услуг;

1. блок контента, который содержит контент раздела;

## область нижнего баннера, в которой дублируются ссылки на основные разделы сайта и на социальные сети, а также указывается контактная

информация и размещается кнопка-ссылка на источник мультимедийного контента сайта.



## Рисунок 2.2 – Дизайн главной страницы

## ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ САЙТА

## PostgreSQL - [свободная](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%9F%D0%9E) [объектно-реляционная](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94) [система управления базами](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) [данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (СУБД).

Существует в реализациях для множества [UNIX-подобных](https://ru.wikipedia.org/wiki/UNIX-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D0%B0%D1%8F_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) платформ, включая [AIX](https://ru.wikipedia.org/wiki/AIX), различные [BSD-системы](https://ru.wikipedia.org/wiki/BSD), [HP-](https://ru.wikipedia.org/wiki/HP-UX)

## [UX](https://ru.wikipedia.org/wiki/HP-UX), [IRIX](https://ru.wikipedia.org/wiki/IRIX), [Linux](https://ru.wikipedia.org/wiki/Linux), [macOS](https://ru.wikipedia.org/wiki/MacOS), [Solaris](https://ru.wikipedia.org/wiki/Solaris)/[OpenSolaris](https://ru.wikipedia.org/wiki/OpenSolaris), [Tru64](https://ru.wikipedia.org/wiki/Tru64), [QNX](https://ru.wikipedia.org/wiki/QNX), а также для [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows).

Важнейшими особенностями PostgreSQL, благодаря которым эта программа приобрела популярность, являются:

1. PostgreSQL – это СУБД с открытым кодом, то есть любой желающий может бесплатно скачать программу на сайте разработчика [(h](http://dev.mysql.com/downloads/))t[tp://dev.mysql.com/downloads/)](http://dev.mysql.com/downloads/)) и при необходимости доработать ее;
2. PostgreSQL имеет развитую систему обеспечения безопасности и разграничения доступа на основе системы привилегий.
3. PostgreSQL — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций.

## Концептуальное проектирование базы данных

## В работе Шацкова [2] сказано, что ER-модель (модель «сущность – связь») – широко используемый инструмент разработки баз данных, представляющий собой формализованное графическое представление предметной области в виде прямоугольников, линий и специальных символов. Преимущество ER-модели заключается в том, что она понятна и разработчикам баз данных, и заказчикам, то есть будущим ее пользователям, и потому все возможные изменения в структуре данных согласуются между ними именно на уровне этой модели. В основе ER-модели лежат понятия

«сущность» («entity») и «связь» («relation»). Не существует удовлетворительного формального описания понятия «сущность». Можно

## сказать, что это – совокупность значимых объектов, существующих в рамках предметной области. Каждая сущность включает набор атрибутов, представляющих собой именованные свойства сущности. Между сущностями могут быть установлены связи, определяющие, как сущности соотносятся или взаимодействуют между собой. Следовательно, для построения ER-модели необходимо сначала выявить сущности, характерные для заданной предметной области, и определить для каждой сущности набор атрибутов, имеющих значение с учетом поставленной задачи. Затем необходимо определить все имеющиеся связи между сущностями и характеристики этих связей. После этого строится ER-диаграмма. Для простых случаев ER- диаграмму можно рисовать вручную, для более сложных можно воспользоваться специализированным программным обеспечением, известным как CASE-средства. В данной была концептуальная модель была построена с использованием программного продукта Microsoft Visio 2016. Концептуальная модель представлена на рисунке 3.1.

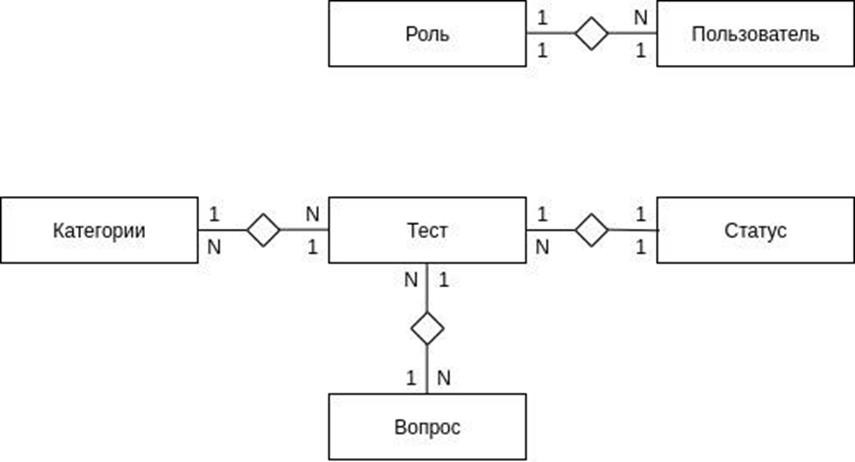


Рисунок 3.1 - Концептуальная модель базы данных

## Логическое проектирование базы данных

Второй этап проектирования базы данных состоит в создании логической модели данных для исследуемой части предприятия.

## Концептуальная модель данных, созданная на предыдущем этапе, уточняется и преобразуется в логическую модель данных.

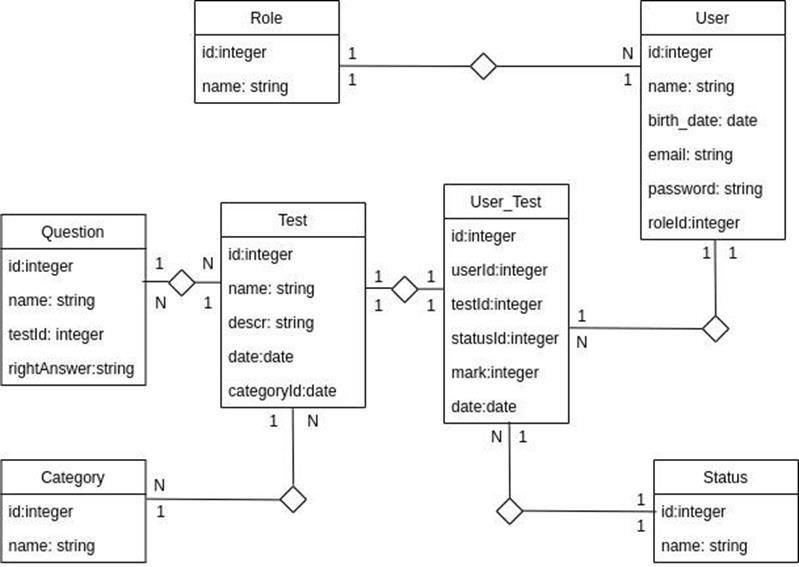
В работе Черноморовой [3] сказано, что на этапе логического моделирования осуществляется нормализация данных, если в этом есть необходимость, выявляются дочерние и родительские отношения, определяются первичные, потенциальные, внешние и родительские ключи.

## Для некоторых столбцов требуется наличие в каждой строке таблицы конкретного и допустимого значения, отличного от NULL. Для задания ограничения подобного типа стандарт SQL предусматривает использование спецификации NOT NULL.

Первичный ключ для каждой строки таблицы должен быть уникальным и иметь непустое значение в каждой строке, что поддерживается с помощью фразы PRIMARY KEY NOT NULL. Для потенциальных ключей используется ключевое слово UNIQUE. Потенциальный ключ может иметь пустое значение, поэтому фраза NOT NULL отсутствует.

## Для обеспечения ссылочной целостности стандарт SQL предусматривает механизм определения внешних ключей с помощью предложения FOREIGN KEY, а фраза REFERENCES определяет имя родительской таблицы. При использовании этого предложения система отклонит выполнение таких DML-операций, как INSERT или UPDATE, с помощью которых будет предпринята попытка создать в дочерней таблице значение внешнего ключа, не соответствующее одному из уже существующих значений первичного или потенциального ключа родительской таблицы.

Логическая модель данных учитывает особенности выбранной модели организации данных в целевой СУБД. Логическая модель, построенная с использованием на рисунке 3.2.



## Рисунок 3.2 - Логическая модель базы данных

## Физическая модель базы данных

## Физическое проектирование - определение особенностей хранения данных, методов доступа и остальных атрибутов базы данных. На этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

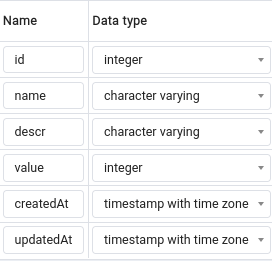
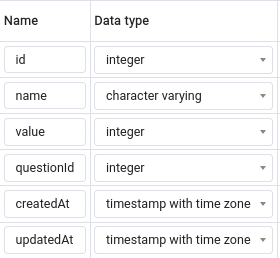


Рисунок 3.3 – Физическая модель таблицы ТипыДолжностей



## Рисунок 3.4 – Физическая модель данных таблицы Ответ

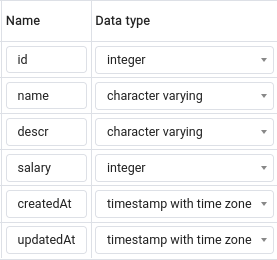
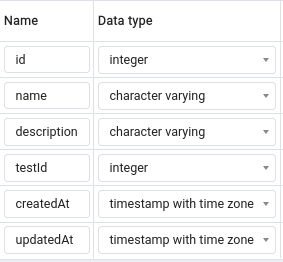


Рисунок 3.5– Физическая модель данных таблицы Должности



## Рисунок 3.6– Физическая модель данных таблицы Вопрос

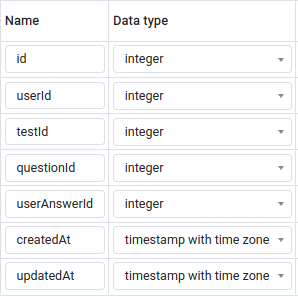
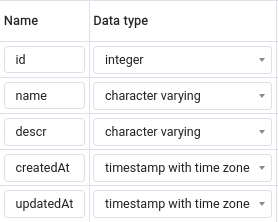


Рисунок 3.7– Физическая модель данных таблицы РезультатыТеста



## Рисунок 3.8– Физическая модель данных таблицы Тест

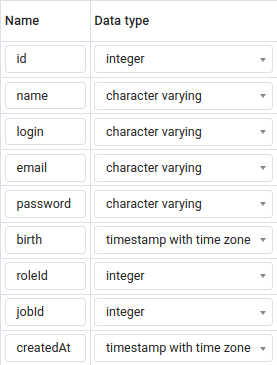


Рисунок 3.9 – Физическая модель данных таблицы Пользователь

## ПРОГРАММНАЯ РЕАЛИЗАЦИЯ САЙТА

* 1. Функциональная структура и модули сайта Функциональное наполнение сайта предусматривает:

## просмотр записей

* создание записей

## прохождение тестов

* регистрация и авторизация

## В данном контексте подразумеваются записи базы данных, такие как:

* Работники

## Категории тестов

* Отделы предприятия

## Сотрудники

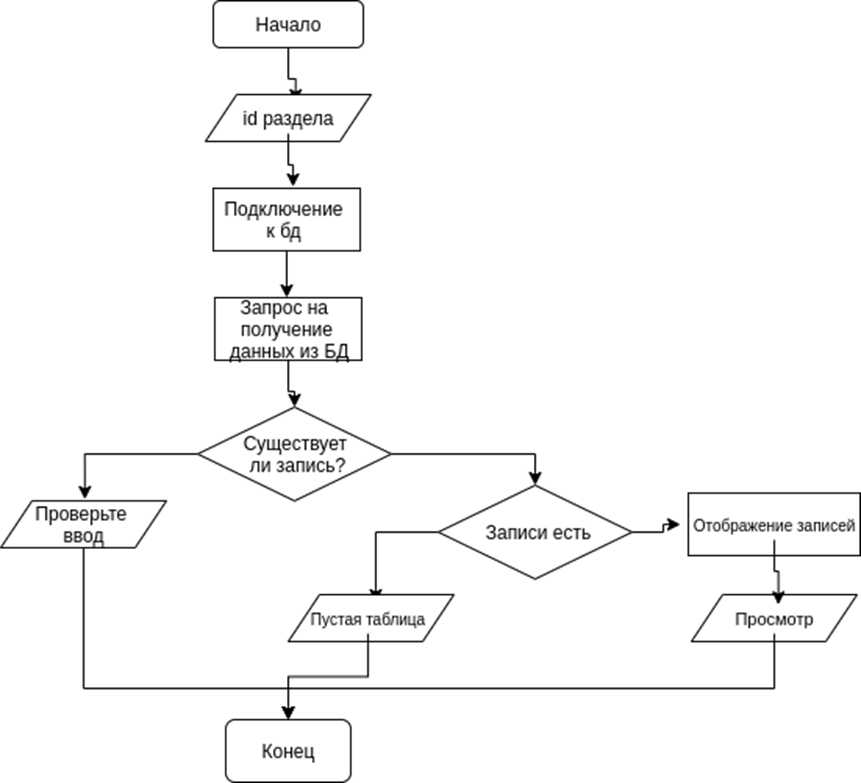
* Тесты

## Вопросы

* Отчеты

## В последующем из представленных в базе данных вопросов и ответов будут складываться тесты для подтверждения квалификации.

Для просмотра записей используется алгоритм, представленный на рисунке 4.1:



## Рисунок 4.1 — Алгоритм получения и просмотра записей из базы данных

## Описание пользовательского интерфейса

## Что касается самого пользовательского интерфейса, то, как я уже говорила, Web-сайт имеет следующие Web-страницы: “Главная”,

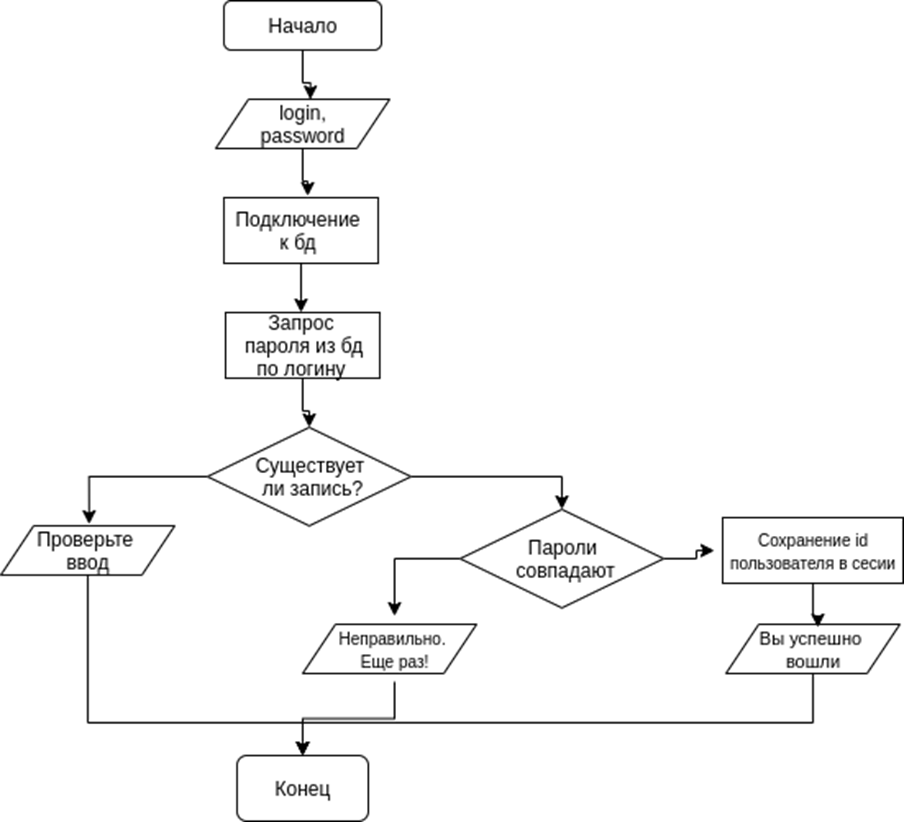
“Категории”, “Отделы”, “Сотрудники”, “Должности”, “Вход”. Пользователь может просматривать каждую из страниц, выбрав в категории тестов

## необходимый, он сможет пройти этот тест и получить за него оценку (подводятся результат тестирования).

Так же доступно следующее взаимодействие: на сайте пользователь может пройти идентификацию, сделать это он сможет на Web-странице “Войти”, “войдя в систему”, пользователь сможет получить результаты тестирования по почте.

## Для авторизации используется алгоритм, представленный на рисунке

4.2:



## Рисунок 4.2 — Алгоритм авторизации пользователя

## Реализации админ-панели

## Любые материалы (статьи, опросы, фото, видео) попадают на сайт через систему управления контентом. Другими словами, это называется – панель администратора.

Мой сайт, как и любой другой должен постоянно пополняться новой актуальной информацией. Желательно с разметкой, где указано время обновления материалов сайта. Переход в режим администратора у меня

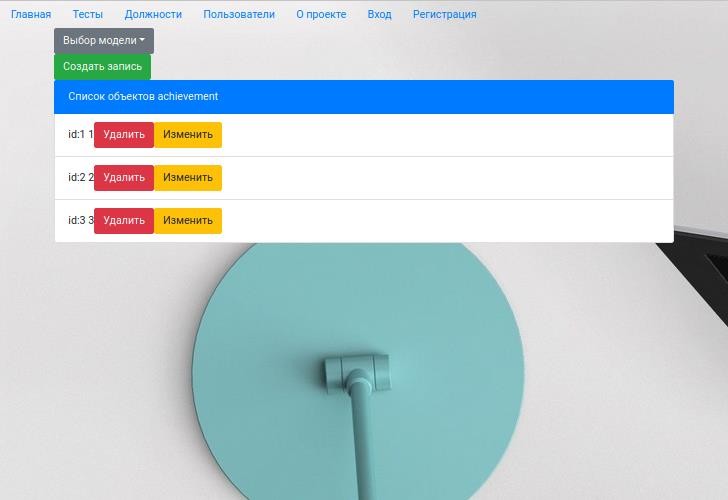
## осуществляется следующим образом: локальный адрес сайта меняется путем

приписывания “\admin” к основному адресу (Рисунок 4.3).



## Рисунок 4.3 – переход в режим администратора

Режим администратора дает возможность пользователю просматривать, а так же добавлять записи в таблицы базы данных. Отмечу то, что дата любых изменений будет автоматически отображаться в моей СУБД. Админ-панель представлена на рисунке 4.4.



## Рисунок 4.4 – админ-панель

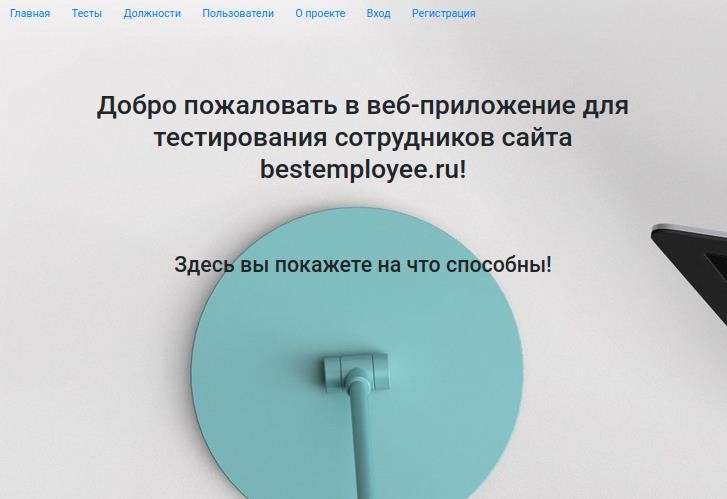
1. ОЦЕНКА ЭФФЕКТИВНОСТИ И ТЕСТИРОВАНИЕ САЙТА

## Для оценки эффективности сайта необходимо произвести расчеты релевантности страниц сайта. В процессе тестирования необходимо будет рассчитать скорость загрузки наиболее нагруженной материалом страницы.

## Тестирование сайта

## При первом посещении пользователь попадает на главную страницу сайта. Данная страница продемонстрирована на рисунке 5.1.

Рисунок 5.1 — Главная страница сайта



## Рисунок 5.2 — Продолжение главной страницы сайта

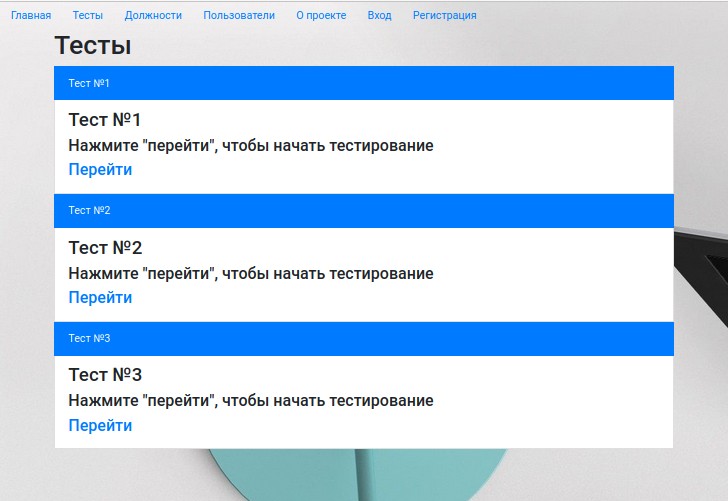
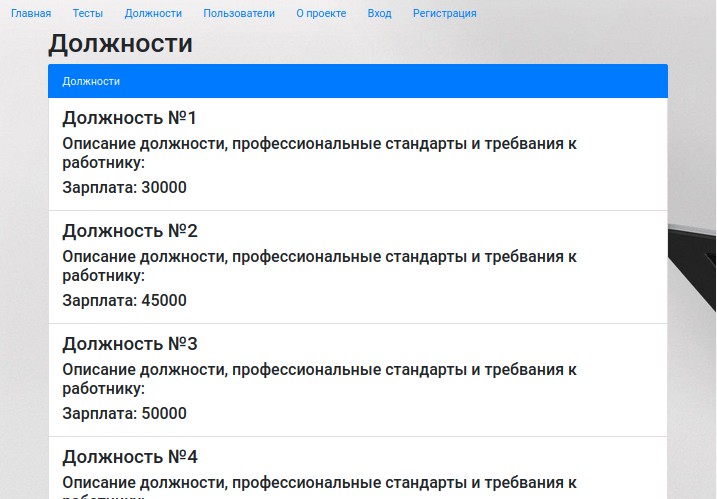


Рисунок 5.3 — Раздел «Тесты» с тестовым списком



## Рисунок 5.4 — Раздел «Должности» с тестовыми записями

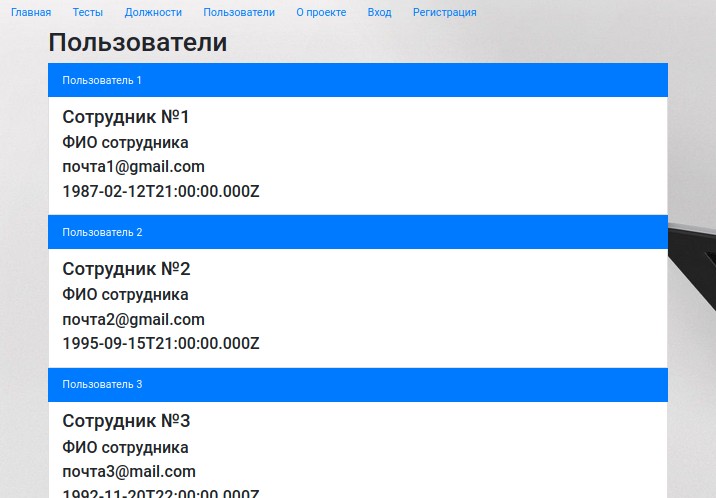
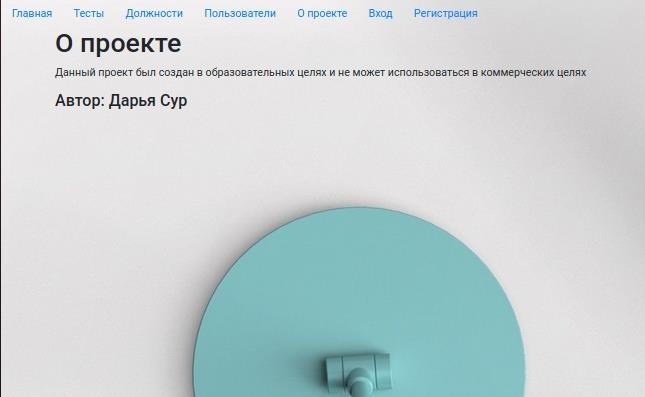


Рисунок 5.5— Страница авторизированных пользователей



## Рисунок 5.6 — Страница с информацией о проекте

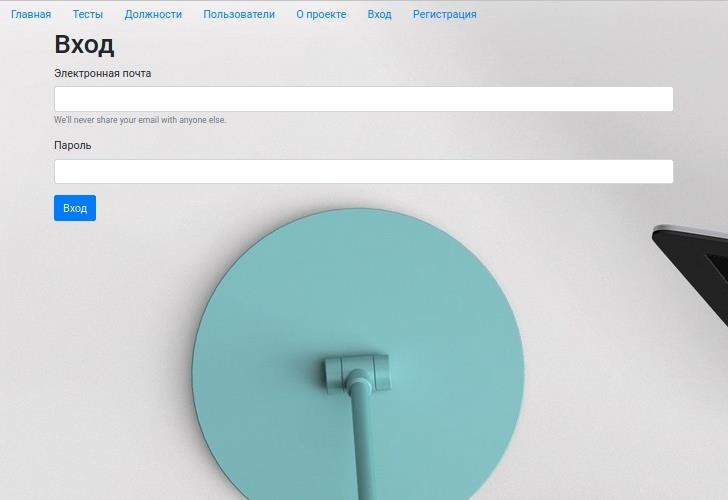
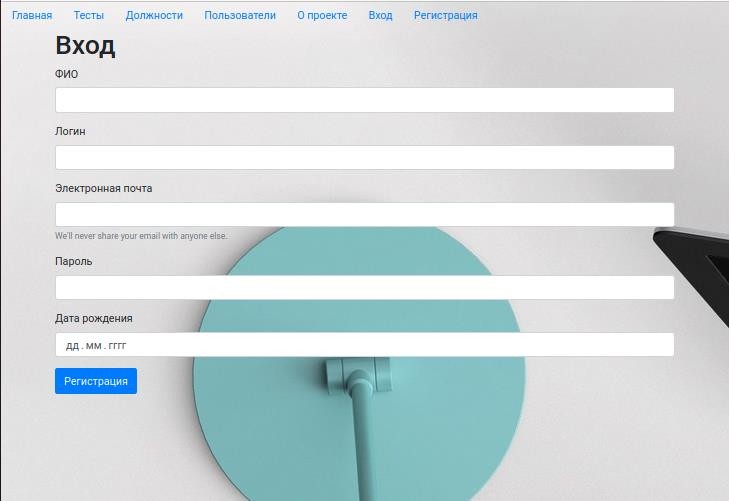


Рисунок 5.7 — Страница с авторизацией пользователя



## Рисунок 5.8 — Страница с регистрацией пользователя

ЗАКЛЮЧЕНИЕ

## Целью данной курсовой работы являлось создание Web-сайта подразделения оценки профессиональных компетенций сотрудников предприятия. Данная цель была достигнута.

В ходе выполнения курсовой работы были реализованы следующие функции:

## регистрация и авторизация пользователя;

* вывод списка тестов;

## вывод списка категорий тестов;

* вывод списка сотрудников

## вывод списка категорий тестов;

* вывод контактной информации;

## вывод информации о тесте;

* вывод информации о сотруднике

## подсчет результатов тестирования.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

## В. Гольцман MySQL 5.0 Библиотека программиста.-М., 2009 – 256

с.

## Проектирование реляционных баз данных: Метод. указания по

выполнению лабораторной работы по курсу “Базы данных” для студентов специальности 230400 “Прикладная математика”/ СПбГАСУ; Сост.: В. В. Шацков СПб., 2009. 21 с.

## Базы данных : учебно-методическое пособие по выполнению курсовой работы для студентов бакалавриата направлений подготовки

«Информационные системы и технологии», «Прикладная информа- тика» / Т.С. Черноморова; Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова. − Новочеркасск: ЮРГПУ (НПИ), 2017. − 24 с.

## Панфилов А.Н. Разработка приложений для Интернет: методические указания по выполнению курсовой работы/ Панфилов Южно- Российский государственный политехнический университет (НПИ) имени М.И.Платова.- Новочеркасск: Лик,2016,-32с,

1. А.Н.Панфилов, В.А.Зуев, А.Н.Скоба Разработка приложений для Интернет: методические указания к выполнению лабораторных работ/ Южно- Российскийгосударственный политехнический университет (НПИ) имени М.И. Платова. - Новочеркасск: Лик,2016.-12 с.

## Приложение А. Техническое задание на разработку Web-Сайта

Техническое задание на разработку Web-Сайта

### ОБЩИЕ ПОЛОЖЕНИЯ

* 1. **Полное наименование системы и ее условное обозначение:** Web-сайт для коммерческой организации “Лучший работник” **Краткое наименование системы:** ЛучшийРаботник.ру

**Имя сайта (название домена):** BestEmployee.ru

### Наименование организации-заказчика и организаций-участников работ

Заказчиком системы является ООО Рекламное агентство “Best”

### a. Перечень документов, на основании которых создается система

Основанием для разработки является типовой договор на разработки Web- проекта.

**1.6 Плановые сроки начала и окончания работы по созданию системы** Плановый срок начала работ по созданию – 10 сентября 2020 года. Плановый срок окончания работ - 25 декабря 2020 года.

### a. Порядок оформления и предъявления заказчику результатов работ по созданию системы

Система передается в виде функционирующего комплекса на базе средств вычислительной техники Заказчика и Исполнителя в сроки, установленные Договором. Приемка системы осуществляется комиссией в составе уполномоченных представителей Заказчика и Исполнителя.

### 1.8 Перечень нормативно-технических документов, методических материалов, использованных при разработке ТЗ

При разработке автоматизированной системы и создании проектно- эксплуатационной документации Исполнитель должен руководствоваться требованиями следующих нормативных документов:

1. ГОСТ 34.601-90. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания;
2. ГОСТ 34.201-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Виды, комплексность и обозначение документов при создании автоматизированных систем;
3. РД 50-34.698-90. Методические указания. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.

### НАЗНАЧЕНИЯ И ЦЕЛИ СОЗДАНИЯ САЙТА

* 1. **Цели создания системы**

Сайт создан в целях ведения учета о квалификации сотрудников данного предприятия, а так же для онлайн подтверждения квалификации для той или иной должности. Так же система несет информативный характер (главная и исчерпывающая информация должны быть расположены на главной странице).

Примечание: сайт не должен содержать рекламу фирмы-заказчика, так как он не несет рекламный характер, а нужен для внутрикорпоративного учета.

### Цели создания системы

Сайт позволяет установить эффективность рабочих кадров, а так же вести учет о успеваемости в работе для каждого сотрудника. Другими словами, данная система должна повысить уровень и качество работы внутри фирмы-заказчика.

### ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

В базе данных сайта должны содержаться таблицы с информацией о сотрудниках, сайт должен содержать информативные статьи должен предоставлять онлайн тестирование для определения оценки знаний в для той или иной должности.

### ТРЕБОВАНИЯ К СИСТЕМЕ

* 1. **Язык сайта**

Русский и английский

### Объем и состав текстовой информации

Ограничение на объем текстовой информации на сайте отсутствует. Что касается состава текстовой информации: должны быть отражена основная информация по соответствующей теме.

### Ключевые слова, по которым сайт должны находить по запросам в поисковых системах и Интернет – каталогах

“Пользователь”, “Сайт”, “Квалификация”, “Оценка”, “Персонал”, “Тестирование”, “Управление”, “Менеджер”, “Директор”, “Рекламное агентство”, “Продажи”, “Развитие”, “javascript”, “php”, “sql”, “mysql”, “Ответственность”, “Подразделения”, “Профессионализм”.

### Объем и состав графической информации

Количество графической информации на сайте не ограничено.

### 4.5 Общий объём и состав текстовой и графической информации, переданный Заказчиком в электронном виде

Отсутствует

### 1.1. Предполагаемая возрастная аудитория сайта

От 21 года и старше.

### Количество страниц сайта

Сайт должен содержать следующие отдельные html страницы: 1-главная (домашняя страница), 2-страница “О сайте”, 3-страница “Регистрация”, 4-страница “Тест”, данные страницы должны являться основными и обязательными.

Примечание: на страницах “О сайте”, “Регистрация”, “Тест” и главной станице могут быть размещены ссылки на второстепенные страницы сайта (например, ссылки на страницы с тестами). При этом ссылки на данные страницы не должны быть расположены в секции с главными ссылками.

### Кнопки управления (навигация сайта):

С каждой страницы сайта должен быть обеспечен переход (установлены гиперссылки) на все страницы сайта, эти ссылки должны находиться в отдельной секции, схематично это выглядит так (Рис.1)



Рис.1

### 4.9 Схема сайта

Главная (начальная) страница сайта должна содержать гиперссылки, обеспечивающие переход с нее на не менее чем 70 % страниц сайта, но не более чем 10 гиперссылок. Так же (Рис.2)

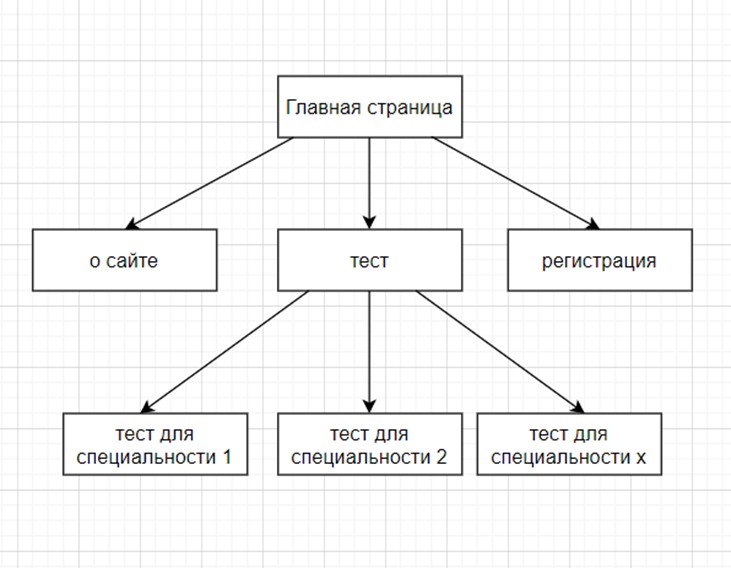


Рис.2-схема сайта

### 1.1 Общий объем сайта, Мб

Предела не имеет.

### 4.11 Оформление рисунков

Формат всех рисунков \*.gif или \*.jpg

### 1.1 Пропускная способность линии связи

Среднее время загрузки страниц не должно превышать 1 секунды при скорости соединения от 200 Кбит/с. Головная (начальная) страница должна иметь время загрузки не более 1,5 секунды. Допускается увеличение времени загрузки отдельных страниц до 36 секунд, но не более чем на 30% общего числа страниц сайта

### Основной диапазон разрешения мониторов, на которых будет просматриваться сайт

От 1366х768 до 1920х1080 пикселей (от 15,6" до 18,5" LCD).

### Минимальное разрешение монитора, на котором будет просматриваться

**сайт**

1366 х 768 пикселей (15,6").

Примечание: при указанном разрешении должна быть обеспечена возможность

просмотра страниц сайта без горизонтальной прокрутки браузера.

### Основной браузер, которым будет просматриваться сайт, и его минимальная версия.

Яндекс браузер версия 19.10.2 и выше

### Цветовая палитра

Основной режим мониторов, на которых будет просматриваться сайт: 15 разрядов цветов и выше (число цветов 65536 и выше). При разработке сайта должен быть обеспечена возможность его просмотра при использовании безопасной цветовой палитры (разрядность цветов 8). Цветовая палитра сайта должна быть следующей:

#FFFFFF – цвет текста ссылок в головной части главной страницы сайта

#ADD8E6 – данным цветом должны выделяться ссылки при наведении на них курсора

#DCDCDC – фон для названия сайта (помимо данного цветового решения фон должен иметь прозрачность 0.9)

#FFFAFA – фон для поля регистрации пользователей на странице “Регистрация”.

#E9967A - цвет текста ссылок в головной части сайта на html-странице “Регистрация”

#E5E5E5 – фон для информативной надписи на html-странице “О сайте”

Примечание: текст с главной информацией на сайте должен находиться на белом фоне либо цветовом фоне с прозрачностью не менее чем 0.6.

### Общий фон сайта

Общий фон сайта светлый белый - #FFFFFF. На отдельных html-страницах допускается использование фонового рисунков, но это необязательное условие.

### Размер и вид шрифта сайта

Размер шрифта сайта должен быть в пределах 10-18 для оформления текста. Для заголовков используются стандартные размеры шрифта, предусмотренные заголовками в языке HTML. Необходимо использование данных шрифтов - 'Montserrat', ‘ Tahoma’, они и являются основными для всех страниц сайта.

### Срок разработки сайта

Три недели со дня зачисления 100 % предоплаты на расчётный счёт Исполнителя.

### Порядок передачи сайта

Исполнитель передает сайт Заказчику в электронном виде либо по протоколу ftp. Заказчик обязан проверить наличие грамматических и орфографических ошибок на сайте в течение трех рабочих дней и передать информацию об ошибках разработчику. Обнаруженные ошибки Исполнитель обязан устранить в течение трех рабочих дней.

### Дополнительные условия

* Главная страница должна содержать полное наименование системы.
* У всех страниц сайта должна быть одинаковая стилистика.

## Приложение Б. Каскадные таблицы стилей

body {

background-image: url(./public/img/background.png);

}

.nav > li > a { color: white;

}

#banner {

padding-top: 15%;

padding-bottom: 15%;

}

#description { width: 800px; height: 300px;

background-image: url(./public/img/employees.gif);

}

#description > center > p { padding-top: 5%;

font-size: 20px;

}

#teamWork {

width: 800px; height: 1000px;

background-image: url(./public/img/employees-2.gif);

}

#teamWork > center > p { padding-top: 15%; font-size: 27px;

}

#aloneWork {

width: 800px;

height: 300px;

background-image: url(./public/img/employees-3.gif);

}

#aloneWork > center > p { color: white; padding-top: 45%; font-size: 20px;

}

#footer {

background-color: black; width: 800px;

height: 350px;

}

#footer > ul {

padding-top: 150px;

}

#footer > ul > li > a { color: white;

font-style: none; font-size: 17px;

}

#auth-form {

margin-bottom: 20%;

}

Приложение В. Тексты программ

# СSS

body {

background-image: url('./img/background.jpg');

}

#greeting {

padding-top: 10%;

}

#plus {

width: 200px; height: 200px;

}

## JS

import logo from "./logo.svg"; import "./App.css";

import { BrowserRouter as Router, Switch, Route, Link } from "react-router- dom";

import "../node\_modules/bootstrap/dist/css/bootstrap.min.css"; import "../node\_modules/bootstrap/dist/js/bootstrap.min.js"; import About from "./Components/about";

import Jobs from "./Components/jobs"; import Main from "./Components/main"; import Tests from "./Components/tests"; import Users from "./Components/users"; import Signin from "./Components/signin"; import Signup from "./Components/signup";

import SingleTestFunc from "./Components/single-test-func"; import Admin from "./Components/admin/admin";

import Result from "./Components/result";

function App() { return ( <Router>

<div className="App">

<ul class="nav">

<li class="nav-item">

<Link class="nav-link" to="/">

Главная

</Link>

</li>

<li class="nav-item">

<Link class="nav-link" to="/test">

Тесты

</Link>

</li>

<li class="nav-item">

<Link class="nav-link" to="/job">

Должности

</Link>

</li>

<li class="nav-item">

<Link class="nav-link" to="/user"> import { Component } from 'react';

export default class Admin extends Component{ render(){

return <div class='container'>

<h1>Административная панель</h1>

</div>;

}

}

import { Component } from "react";

import { Switch, Route, Link } from "react-router-dom"; import axios from "axios";

import List from "./list";

import Achievement from "./create/achievement";

import AchievementType from "./create/achievementType"; import Answer from "./create/answer";

import Job from "./create/job";

import Question from "./create/question"; import Right from "./create/right"; import Role from "./create/role";

import Test from "./create/test"; import User from "./create/user";

import UserAchievement from "./create/userAchievement";

export default class Admin extends Component { constructor(props) {

super(props);

this.state = { model: "",

};

}

render() {

let list = [ "achievement", "achievementType", "answer",

"job",

"photo", "question", "role",

"test",

"user", "userachievement", "right",

];

let items = list.map((item) => (

<Link class="dropdown-item" to={"/admin/" + item}>

{item}

</Link>

));

let links = list.map((item) => (

<Route exact path={"/admin/" + item}>

<List item={item} />

</Route>

));

return (

<div class="container">

<div class="dropdown">

<a

class="btn btn-secondary dropdown-toggle" href="#"

role="button" id="dropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false"

>

Выбор модели

</a>

<div class="dropdown-menu" aria-labelledby="dropdownMenuLink">

{items}

</div>

<Switch>{links}

<Route exact path="/admin/achievement/create"><Achievement/></Route>

<Route exact

path="/admin/achievementType/create"><AchievementType/></Route>

<Route exact path="/admin/answer/create"><Answer/></Route>

<Route exact path="/admin/job/create"><Job/></Route>

<Route exact path="/admin/question/create"><Question/></Route>

<Route exact path="/admin/right/create"><Right/></Route>

<Route exact path="/admin/role/create"><Role/></Route>

<Route exact path="/admin/test/create"><Test/></Route>

<Route exact path="/admin/user/create"><User/></Route>

<Route exact

path="/admin/userachievement/create"><UserAchievement/></Route>

</Switch>

</div>

</div>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class Achievement extends Component { constructor(props) {

super(props); this.state = {

types: [],

name: "", description: "", type: 0,

};

this.handleChangeName = this.handleChangeName.bind(this); this.handleChangeDescription = this.handleChangeDescription.bind(this); this.handleChangeType = this.handleChangeType.bind(this);

this.handleSubmit = this.handleSubmit.bind(this);

}

handleChangeName(event) {

this.setState({ name: event.target.value });

}

handleChangeDescription(event) {

this.setState({ description: event.target.value });

}

handleChangeType(event) {

this.setState({ type: event.target.value });

}

handleSubmit(event) { axios.post("http://localhost:4000/admin/achievement",

{

id:null,

name: this.state.name,

descr: this.state.description, achievementTypeId: this.state.type

})

.then((result) => console.log(result)); console.log(this.state); event.preventDefault();

}

componentWillMount() { axios

.get("http://localhost:4000/admin/achievementType/")

.then((result) => { this.setState({ types: result.data}) })

.catch((err) => console.log(err));

}

render() {

let list = this.state.types;

let items = list.map((item) => (

<option id={item.id} class="dropdown-item">{item.id}</option>

));

return (

<form onSubmit={this.handleSubmit}>

<label>

Имя:

<input

type="text" onChange={this.handleChangeName}

/>

</label>

<br />

<label>

Описание:

<input

type="text" onChange={this.handleChangeDescription}

/>

</label>

<br />

<label>

Выбор типа достижения:

<select

class="btn btn-primary dropdown-toggle" onChange={this.handleChangeType}

>

{items}

</select>

</label>

<br />

<input type="submit" value="Отправить" />

</form>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class AchievementType extends Component { constructor(props) {

super(props); this.state = {

name: "", description: "", value: null,

};

this.handleChangeName = this.handleChangeName.bind(this); this.handleChangeDescription = this.handleChangeDescription.bind(this); this.handleChangeType = this.handleChangeType.bind(this); this.handleSubmit = this.handleSubmit.bind(this);

}

handleChangeName(event) {

this.setState({ name: event.target.value });

}

handleChangeDescription(event) {

this.setState({ description: event.target.value });

}

handleChangeType(event) {

this.setState({ value: event.target.value });

}

handleSubmit(event) { axios.post("http://localhost:4000/admin/achievementType/",

{

id:null,

name: this.state.name,

descr: this.state.description, value: this.state.value

})

.then((result) => console.log(result)); event.preventDefault();

console.log(this.state);

}

render() { return (

<form onSubmit={this.handleSubmit}>

<label>

Имя:

<input

type="text" onChange={this.handleChangeName}

/>

</label>

<br />

<label>

Описание:

<input

type="text" onChange={this.handleChangeDescription}

/>

</label>

<br />

<label>

Значение

<input

type='number' onChange={this.handleChangeType}

/>

</label>

<br />

<input type="submit" value="Отправить" />

</form>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class Achievement extends Component { constructor(props) {

super(props); this.state = {

types: [],

name: "",

value: "", questionId: ''

};

this.handleChangeName = this.handleChangeName.bind(this); this.handleChangeDescription = this.handleChangeDescription.bind(this); this.handleChangeType = this.handleChangeType.bind(this); this.handleSubmit = this.handleSubmit.bind(this);

}

handleChangeName(event) {

this.setState({ name: event.target.value });

}

handleChangeDescription(event) { this.setState({ value: event.target.value });

}

handleChangeType(event) { console.log(event.target.value);

this.setState({ questionId: event.target.value }); console.log(this.state.questionId);

}

handleSubmit(event) { event.preventDefault();

axios.post("http://localhost:4000/admin/answer/",

{

id:null,

name: this.state.name, value: this.state.value,

questionId: this.state.questionId

})

.then((result) => console.log(result)); event.preventDefault(); console.log(this.state);

}

componentWillMount() { axios

.get("http://localhost:4000/admin/question/")

.then((result) => { this.setState({ types: result.data}) })

.catch((err) => console.log(err));

}

render() {

let list = this.state.types;

let items = list.map((item) => (

<option class="dropdown-item">{item.id}</option>

));

return (

<form onSubmit={this.handleSubmit}>

<label>

Имя:

<input

type="text"

onChange={this.handleChangeName}

/>

</label>

<br />

<label>

Вес:

<input

type="text" onChange={this.handleChangeDescription}

/>

</label>

<br />

<label>

Выбор вопроса:

<select

class="btn btn-primary dropdown-toggle" onChange={this.handleChangeType}

>

{items}

</select>

</label>

<br />

<input type="submit" value="Отправить" />

</form>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class Job extends Component { constructor(props) {

super(props); this.state = {

types: [],

name: "",

descr: "", salary: 0

};

this.handleChangeName = this.handleChangeName.bind(this); this.handleChangeDescription = this.handleChangeDescription.bind(this);

this.handleChangeSalary= this.handleChangeSalary.bind(this); this.handleSubmit = this.handleSubmit.bind(this);

}

handleChangeName(event) {

this.setState({ name: event.target.value });

}

handleChangeDescription(event) { this.setState({ descr: event.target.value });

}

handleChangeSalary(event) {

this.setState({ salary: event.target.value });

}

handleSubmit(event) { event.preventDefault();

axios.post("http://localhost:4000/admin/job/",

{

id:null,

name: this.state.name, descr: this.state.descr, salary: this.state.salary

})

.then((result) => console.log(result)); event.preventDefault(); console.log(this.state);

}

render() {

let list = this.state.types; return (

<form onSubmit={this.handleSubmit}>

<label>

Имя:

<input

type="text" onChange={this.handleChangeName}

/>

</label>

<br />

<label>

Вес:

<input

type="text" onChange={this.handleChangeDescription}

/>

</label>

<br />

<label>

Зарплата:

<input

type="text" onChange={this.handleChangeSalary}

/>

</label>

<br />

<input type="submit" value="Отправить" />

</form>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class Question extends Component { constructor(props) {

super(props); this.state = {

types: [],

name: "", description: "", testId: ''

};

this.handleChangeName = this.handleChangeName.bind(this); this.handleChangeDescription = this.handleChangeDescription.bind(this); this.handleChangeType = this.handleChangeType.bind(this); this.handleSubmit = this.handleSubmit.bind(this);

}

handleChangeName(event) {

this.setState({ name: event.target.value });

}

handleChangeDescription(event) {

this.setState({ description: event.target.value });

}

handleChangeType(event) {

this.setState({ testId: event.target.value });

}

handleSubmit(event) { event.preventDefault();

axios.post("http://localhost:4000/admin/question/",

{

id:null,

name: this.state.name,

description: this.state.description, testId: this.state.testId

})

.then((result) => console.log(result));

}

componentWillMount() { axios

.get("http://localhost:4000/admin/test/")

.then((result) => { this.setState({ types: result.data}) })

.catch((err) => console.log(err));

}

render() {

let list = this.state.types;

let items = list.map((item) => (

<option id={item.id} class="dropdown-item">{item.id}</option>

));

return (

<form onSubmit={this.handleSubmit}>

<label>

Имя:

<input

type="text" onChange={this.handleChangeName}

/>

</label>

<br />

<label>

Описание:

<input

type="text" onChange={this.handleChangeDescription}

/>

</label>

<br />

<label>

Выбор теста в котором будет вопрос:

<select

class="btn btn-primary dropdown-toggle" onChange={this.handleChangeType}

>

{items}

</select>

</label>

<br />

<input type="submit" value="Отправить" />

</form>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class Right extends Component { constructor(props) {

super(props); this.state = {

quetions:[],

answers:[], questionId: null, answerId: null

};

this.chooseQuestion = this.chooseQuestion.bind(this); this.chooseAnswer = this.chooseAnswer.bind(this); this.handleSubmit = this.handleSubmit.bind(this);

}

chooseQuestion(event) {

this.setState({ questionId: event.target.value });

}

chooseAnswer(event) {

this.setState({ answerId: event.target.value });

}

handleSubmit(event) { event.preventDefault();

axios.post("http://localhost:4000/admin/right/",

{

id:null,

questionId: this.state.questionId, answerId: this.state.answerId

})

.then((result) => console.log(result));

}

componentWillMount() { axios

.get("http://localhost:4000/admin/question/")

.then((result) => { this.setState({ quetions: result.data}) })

.catch((err) => console.log(err));

axios

.get("http://localhost:4000/admin/answer/")

.then((result) => { this.setState({ answers: result.data}) })

.catch((err) => console.log(err));

}

render() {

let list = this.state.types;

let questions = list.map((item) => (

<option id={item.id} class="dropdown-item">{item.id}</option>

));

let answers = list.map((item) => (

<option id={item.id} class="dropdown-item">{item.id}</option>

));

return (

<form onSubmit={this.handleSubmit}>

<label>

Выбор вопроса :

<select

class="btn btn-primary dropdown-toggle" onChange={this.chooseQuestion}

>

{ questions }

</select>

</label>

<br />

<label>

Выбор ответа:

<select

class="btn btn-primary dropdown-toggle" onChange={this.chooseAnswer}

>

{ answers }

</select>

</label>

<br />

<input type="submit" value="Отправить" />

</form>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class Role extends Component { constructor(props) {

super(props); this.state = {

name: ''

};

this.handleName = this.handleName.bind(this); this.handleSubmit = this.handleSubmit.bind(this);

}

handleName(event) {

this.setState({ questionId: event.target.value });

}

handleSubmit(event) { event.preventDefault();

axios.post("http://localhost:4000/admin/role/",

{

id:null,

name: this.state.name

})

.then((result) => console.log(result));

}

render() { return (

<form onSubmit={this.handleSubmit}>

<h1>Создайте роль:</h1>

<input type='text' onChange={ this.handleName } placeholder='Введите название роли'/>

<input type="submit" value="Отправить" />

</form>

);

}

}

import { Component } from "react"; import axios from "axios";

export default class List extends Component { constructor(props) {

super(props);

this.state = { instance: [],

};

}

componentWillUpdate() { axios

.get("http://localhost:4000/admin/" + this.props.item)

.then((result) => this.setState({ instance: result.data }))

.catch((err) => console.log(err));

}

deleteItem(e){

let id = e.target.id;

let model = this.props.item;

axios.delete('http://localhost:4000/admin/'+model+'/'+id)

.then((res) => console.log(res))

.catch((err) => console.log(err));

}

render() {

let items = this.state.instance; let list = items.map((item) => (

<li class="list-group-item"> id:{item.id} {item.name}

<button type="button" id={item.id} class="btn btn-danger"

onClick={this.deleteItem}>Удалить</button>

<button type="button" id={item.id} class="btn btn-warning"

onClick={this.updateItem}>Изменить</button>

</li>

));

return (

<div>

<button type="button" class="btn btn-success"

onClick={this.createItem}>Создать запись</button>

<ul class="list-group">

<li class="list-group-item active">

Список объектов {this.props.item}

</li>

{ list }

</ul>

</div>

);

}

}

import { Component } from "react"; import axios from "axios";

import Question from "./question";

import { Switch, Route, Link } from 'react-router-dom'; import Result from './result';

export default class SingleTest extends Component { constructor(props) {

super(props);

this.state = { questions: [],

answers: [], userAnswers: [],

};

this.getAnswer = this.getAnswer.bind(this);

}

getAnswer(e) {

this.setState({ userAnswers: { ...this.state.userAnswers, [e.target.name]:e.target.value}});

}

getResult(e){ axios

.post('http://localhost:4000/test/check', this.state.userAnswers)

.then((result) => console.log(result))

.catch((err) => console.log(err));

e.preventDefault();

}

componentDidUpdate() { console.log(this.state);

}

componentWillMount() {

//получить список вопросов

axios

.get("http://localhost:4000/getQuestions/" + this.props.testId)

.then((result) => this.setState({ questions: result.data }))

.catch((err) => console.log(err));

}

render() {

let questions = this.state.questions; let list = questions.map((question) => (

<Question setAnswer={this.getAnswer.bind(this)} question={question} />

));

return (

<div class="container">

<form onChange={this.getAnswer} onSubmit={this.getResult}>

{list}

</form>

<Result/>

</div>

);

}

}